AF
JFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.: 09/739,618
Filed: December 18, 2000
Inventor(s):
John H. Howard

Title: Object-Based Storage Device with Improved Reliability and Fast Crash Recovery

§ § § § § § § § § § § § § § §

Examiner: Duong, Thomas
Group/Art Unit: 2145
Atty. Dkt. No: 5181-59100

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below.

Lawrence J. Merkel
Printed Name

Signature 3/13/06 Date

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed December 21, 2005 (with Pre-Appeal Brief Request for Review) and the Pre-Appeal Brief Decision mailed March 1, 2006, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The present application is owned by Sun Microsystems, Inc. An assignment of the present application to Sun Microsystems, Inc. is recorded at Reel 011810, Frame 0381.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences known to Appellant.

III. STATUS OF CLAIMS

Claims 2-10, 12-20, 22-27, 29-35, and 40-44 are pending. Claims 2-10, 12-20, 22-27, and 29-35 are rejected under 35 U.S.C. § 103(a). Claims 40-44 are rejected under 35 U.S.C. § 102(b). It is these rejections that are being appealed. A copy of claims 2-10, 12-20, 22-27, 29-35, and 40-44 is included in the Claims Appendix attached hereto.

IV. STATUS OF AMENDMENTS

No unentered amendment to the claims has been filed after final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 2 is directed to a storage (12A) comprising a non-volatile memory (44, 48) storing a first inode locating a first file in said storage and also storing a journal (70) comprising a list of committed inodes, and a block manager (42). The block manager is configured to copy the first inode to a second inode, to change the second inode in response to updates to the first file, and to atomically update the first file in response to a commit of the first file. The atomic update may be accomplished by writing the second inode to the non-volatile memory, whereby the second inode locates the first file in the storage. The block manager is configured to record the second inode in the journal (See, e.g., Fig. 2 and Fig. 4, and specification page 9, line 26-page 10, line

14; page 15, line 14-page 17, line 24).

Independent claim 8 is directed to an apparatus comprising a computing node (10A) configured to perform one or more write commands to a file and a commit command committing the one or more write commands to the file, and a storage (12A) coupled to receive the write commands and the commit command. The storage is configured to copy one or more blocks of the file (B2, Figs. 10 and 11) to a copied one or more blocks (B2', Figs. 10 and 11), wherein the copied blocks are the blocks updated by the write commands. The storage is configured to update the copied blocks with write data corresponding to the write commands. Additionally, the storage is configured to copy a first inode (164, 176) locating the file in the storage to a second inode (166, 177), and to update pointers within the second inode pointing to the blocks to point to the copied blocks. The storage is configured to atomically update the file by writing the second inode responsive to the commit command. The first inode is stored in an inode file (72) that is identified by a master inode (78A), and the inode file is atomically updated with the second inode by writing the master inode subsequent to the commit command. (See, e.g., Figs. 2, 4, 10, and 11; specification page 9, line 26-page 10, line 14; page 15, line 14-page 17, line 24).

Independent claim 12 is directed to a method. A first inode is copied to a second inode, wherein the first inode locates a first file in a storage (12A). The second inode is modified in response to one or more changes to the first file. The first file is atomically updated by establishing the second inode as the inode for the first file. Establishing the second inode comprises storing the second inode in a journal (70), wherein the journal is stored in a nonvolatile memory. (See, e.g., Figs. 2 and 4; specification page 9, line 26-page 10, line 14; page 15, line 14-page 17, line 24).

Independent claim 22 is directed to a storage (12A) comprising a non-volatile memory (44, 48) storing a first inode locating a first version of a file in said storage and also storing a journal (70) comprising a list of committed inodes. The storage further comprises a block manager (42) configured to copy the first inode to a second inode, to

change the second inode in response to updates to the file, and to atomically update the file in response to a commit of the file. The atomic update produces a second version of the file. The atomic update may be accomplished by writing the second inode to the non-volatile memory, wherein the second inode locates the second version of the file in the storage. The block manager is configured to record the second inode in the journal. (See, e.g., Figs. 2 and 4; specification page 9, line 26-page 10, line 14; page 15, line 14-page 17, line 24).

Independent claim 29 is directed to a method. A first inode is copied to a second inode, wherein the first inode locates a first version of a file in a storage (12A). The second inode is modified in response to one or more changes to the file, creating a second version of the file. The file is atomically updated to the second version by establishing the second inode as the inode for the file, wherein the establishing comprises storing the second inode in a journal (70) stored in a nonvolatile memory (44, 48). (See, e.g., Figs. 2 and 4; specification page 9, line 26-page 10, line 14; page 15, line 14-page 17, line 24).

Independent claim 40 is directed to an apparatus comprising a computing node (10A), an interconnect (20) to which the computing node is coupled, and a storage (12) coupled to the interconnect. The computing node is configured to generate a plurality of write commands to update a first file and a commit command defined to commit the updates to the first file. The computing node is configured to transmit the plurality of write commands and the commit command on the interconnect. The storage is configured to store the first file, and is configured to atomically update the first file to reflect the plurality of write commands responsive to the commit command. (See, e.g., Figs. 1, 2, and 4 and specification page 5, line 4-page 6, line 21; page 9, line 26-page 10, line 14; page 15, line 14-page 17, line 24).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 2-10, 12, 18-20, 22-27, 29, and 35 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Senator et al., U.S. Patent No. 5,761,677 ("Senator") in view of Zheng et al., U.S. Patent No. 6,571,259 ("Zheng").

2. Claims 13-17 and 30-34 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Senator in view of Zheng and Raz, U.S. Patent No. 5,701,480 ("Raz").

3. Claims 40-44 are rejected under 35 U.S.C. § 102(b) as being anticipated by Senator.

VII. ARGUMENT

First Ground of Rejection:

Claims 2-10, 12, 18-20, 22-27, 29, and 35 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Senator in view of Zheng. Appellant traverses this rejection for at least the following reasons.

Claims 2-5, 12, 18-20, 22-25, 29, and 35:

Appellant respectfully submits that each of claims 2, 12, 22, and 29 recites a combination of features not taught or suggested in Senator and Zheng. For example, claim 2 recites a combination of features including: "a non-volatile memory storing a first inode locating a first file in said storage and also storing a journal comprising a list of committed inodes; and a block manager ... configured to atomically update said first file in response to a commit of said first file by writing said second inode to said non-volatile memory ... and wherein said block manager is configured to record said second inode in said journal".

Appellant respectfully submits that Senator in view of Zheng fails to form a *prima facie* case of obviousness, because Senator teaches away from the alleged combination and because the combination fails to teach or suggest each and every feature of the claim.

Firstly, **Senator teaches away** from making the combination. The Final Office

Action mailed October 5, 2005 ("Office Action") relies on Zheng to teach a journal. Senator also discloses, in the background, the existence of journals of changes to file system records (e.g. Senator, col. 1, lines 28-35). However, Senator teaches away from using such journals/data logs in his invention (on which the Office Action relies for other features of claim 2). For example, Senator teaches "The VERSION module 107 provides for various versions of a file without the need for data copy or log operations". (Senator, col. 3, lines 39-40).

In fact, **avoiding the use of such journals**/data logging is a **central feature** of Senator's invention: "Various versions of a computer file are provided without requiring copying the file or logging changed data, so that the files have consistent user data" (Senator, abstract, lines 1-3); and "In accordance with the instant invention, the above problems of inconsistency and throughput bottlenecks have been solved by providing for various versions of a file without the need for data copy or log operations" (Senator, col. 1, lines 63-65). See also Senator, col. 2, lines 13-15; col 5 lines 43-47. In fact, Senator handles the consistency of file data and multiple versions of files in a completely different way, storing pointers to current and previous inodes in the inodes themselves (see, e.g., Senator, Figs 3c-3e, elements 305 and 307 and col. 5, lines 12-25). Even the section of Senator used in the Office Action as evidencing Senator's acknowledgement of journals/data logging teaches away from such a combination: "No separate data logs need be kept for purposes of data consistency" (Senator, col. 2, lines 26-27).

Secondly, the alleged combination **fails to teach or suggest** each and every feature of the claim. The Office Action relies on Zheng to teach a journal. Particularly, the Office Action relies on Zheng's file system index 37 and Zheng's LRU list 36. Zheng's file system index and LRU list are "in-memory". Zheng's teaching further include the following: "All disk-block reservation and pre-allocation mapping are in the memory, and after a crash, they are automatically discarded" (See Zheng, abstract, for example). Thus, Zheng intends for all in-memory information (including the file system index 37 and the LRU list 36) to be automatically discarded in a crash. These teachings do not teach or suggest "a non-volatile memory storing a first inode locating a first file in

said storage and also storing a journal comprising a list of committed inodes". Rather, **the described operation in Zheng is that of volatile memory**. The Response to Arguments section in the Office Action states that the reference does not use the term "volatile memory", and argues that recovering from a system failure by restoring the database to its consistent state existing just after commitment of the last complete transaction indicates that the memory must be non-volatile. Appellant respectfully disagrees. **Zheng clearly describes the memory operating in a volatile fashion.** Furthermore, it would not be obvious to modify Zheng's teachings to store his file system index 37 or LRU list 36 in a non-volatile memory, as it would change the intended purpose of discarding all in-memory information. In summary, storing the file system index 37 and the LRU list 36 in a memory and automatically discarding such data, as taught in Zheng, does not teach or suggest "a non-volatile memory storing a first inode locating a first file in said storage and also storing a journal comprising a list of committed inodes". For at least the above stated reasons, Appellant submits that the rejection of claim 2 over the alleged combination of Senator and Zheng is not supported, and should be withdrawn.

Claim 12 recites a combination of features including: "atomically updating said first file by establishing said second inode as the inode for said first file, wherein said establishing comprises storing said second inode in a journal stored in a nonvolatile memory". The same teachings of Senator and Zheng highlighted above with respect to claim 2 are relied on to reject claim 12. Therefore, the rejection of claim 12 over the alleged combination of Senator and Zheng is also not supported, and should be withdrawn.

Claim 22 recites a combination of features including: "a non-volatile memory storing a first inode locating a first version of a file in said storage and also storing a journal comprising a list of committed inodes; and a block manager ... configured to atomically update the file, producing a second version of the file, in response to a commit of the file by writing said second inode to said non-volatile memory... and wherein said block manager is configured to record said second inode in said journal". The same

teachings of Senator and Zheng highlighted above with respect to claim 2 are relied on to reject claim 22. Therefore, the rejection of claim 22 over the alleged combination of Senator and Zheng is also not supported, and should be withdrawn.

Claim 29 recites a combination of features including: "atomically updating the file to the second version by establishing said second inode as the inode for the file, wherein said establishing comprises storing said second inode in a journal stored in a nonvolatile memory". The same teachings of Senator and Zheng highlighted above with respect to claim 2 are relied on to reject claim 29. Therefore, the rejection of claim 29 over the alleged combination of Senator and Zheng is also not supported, and should be withdrawn.

For at least the above stated reasons, Appellant submits that the rejection of claims 2, 12, 22, and 29 is in error and requests reversal of the rejection. The rejection of claims 3-5 (dependent from claim 2), claims 18-20 (dependent from claim 12), claims 23-25 (dependent from claim 22), and claim 35 (dependent from claim 29) are similarly in error for at least the above stated reasons, and reversal of the rejection is requested. Each of claims 3-5, 18-20, 23-25, and 35 recite additional combinations of features not taught or suggested in the cited art.

Claims 6 and 26:

Claims 6 and 26 depend from claims 2 and 22, respectively. Accordingly, the rejection of claims 6 and 26 is in error for at least the reasons highlighted above with regard to claims 2 and 22. Additionally, each of claims 6 and 26 recite a combination of features including: "said journal further includes a checkpoint record including a description of an inode file, a block allocation bitmap, and an inode allocation bitmap."

The Office Action alleges that the above highlighted features are taught in Senator and Zheng, specifically citing col. 2, lines 13-46 and col. 4, line 45-col. 4, line 47 of Senator and col. 3, line 3-col. 4, line 14 and col. 14, line 46-col. 15, line 14 of Zheng. Appellant respectfully submits that nothing in these sections teaches or suggests the

above highlighted features.

Senator does not teach or suggest a journal, as the Office Action admits with regard to claim 2. Accordingly, Senator cannot teach a checkpoint record in a journal, let alone a checkpoint record that includes a description of an inode file, a block allocation bitmap, or an inode allocation bitmap. Furthermore, the cited sections of Senator teach nothing with regard to a checkpoint record, nor the block allocation bitmap and inode allocation bitmap. While the cited portions of Zheng do generally discuss block allocation and inode allocation, no mention of a checkpoint record, a block allocation bitmap, and an inode allocation bitmap is made.

For at least the above stated reasons, Appellant submits that the rejection of claims 6 and 26 is in error and requests reversal of the rejection.

Claims 7 and 27:

Claims 7 and 27 depend from claims 6 and 26, respectively. Accordingly, the rejection of claims 7 and 27 is in error for at least the reasons highlighted above with regard to claims 6 and 26. Additionally, each of claims 7 and 27 recite a combination of features including: "the description comprises inodes for each of said inode file, said block allocation bitmap, and said inode allocation bitmap."

The same sections of Zheng and Senator highlighted above with regard to claims 6 and 26 are cited to allegedly teach the features of claims 7 and 27. However, these sections do not teach or suggest the block allocation bitmap and the inode allocation bitmap, as highlighted above. Furthermore, Zheng and Senator do not teach or suggest inodes for the block allocation bitmap and the inode allocation bitmap.

For at least the above stated reasons, Appellant submits that the rejection of claims 6 and 26 is in error and requests reversal of the rejection.

Claims 8-10:

Appellant respectfully submits that claim 8 recites a combination of features not taught or suggested in Senator and Zheng. For example, claim 8 recites a combination of features including: "a storage coupled to receive said one or more write commands ... wherein said storage is configured to copy one or more blocks of said file to a copied one or more blocks, said one or more blocks updated by said one or more write commands, and wherein said storage is configured to update said copied one or more blocks with write data corresponding to said one or more write commands".

The Office Action alleges that the combination of Senator and Zheng renders claim 8 obvious, but the rejection is grouped with that of claims 2, 12, 22 and 29 discussed above. The above highlighted features are not treated in that rejection. Accordingly, a **proper *prima facie* case of obviousness of claim 8 has not been established**, because EACH and EVERY feature in claim 8 has not been identified in the prior art.

In the Response to Arguments Section, the Office Action alleges that Senator teaches the above highlighted features at col. 2, lines 13-46 and col. 4, line 45-col. 5, line 48. Appellant respectfully disagrees, and asserts that these sections teach various details of making file updates without copying data from old blocks to newly allocated blocks. For example, the following teachings are illustrative: "a module [is] responsive to a system call argument to allocate another node in the file system tables and to copy the data block allocations from the old node into the newly allocated node. Both nodes now contain the same data block allocation information. Shadow pointers are set in the old node to point to the new node and set in the new node to point to the old node. Changes to the actual data are now made with respect to the new node and fresh data blocks are allocated for the changed blocks." (Senator, col. 2, lines 15-23). Senator includes no teaching of copying data from a previous data block to the newly allocated data block when an update is made. In fact, Senator repeatedly teaches that no file data is copied in his system: "The VERSION module 107 provides for various versions of a file without the need for data copy or log operations". (Senator, col. 3, lines 39-40). Avoiding the

use of such copying is a central feature of Senator's invention: "Various versions of a computer file are provided without requiring copying the file or logging changed data, so that the files have consistent user data" (Senator, abstract, lines 1-3); and "In accordance with the instant invention, the above problems of inconsistency and throughput bottlenecks have been solved by providing for various versions of a file without the need for data copy or log operations" (Senator, col. 1, lines 63-65). See also Senator, col. 2, lines 13-15; col 5 lines 43-47. While Senator copies allocation information from node to node, no copying of data from old blocks to newly allocated blocks is taught in Senator.

For at least the above stated reasons, Appellant submits that the rejection of claim 8 is in error and request reversal of the rejection. Claims 9-10 depend from claim 8, and thus the rejection of claims 9-10 is in error for at least the above stated reasons as well. Claims 9-10 recite additional combinations of features not taught or suggested in Senator and Zheng.

Second Ground of Rejection:

Claims 13-17 and 30-34 are rejected under 35 U.S.C. § 103(a) as being unpatentable over the Senator in view of Zheng and Raz. Appellant traverses this rejection for at least the following reasons.

With respect to all of claims 13-17 and 30-34, Appellant notes that the Office Action admits that Senator and Zheng do not teach or suggest the additional features recited in these claims. The Office Action relies on Raz to allegedly teach ALL such features. However, Raz teaches an after-image log, checkpointing, and recovery of a multi-version database. The records of the database are not files, and the checkpointing of record updates and other features of Raz's database do not teach or suggest inodes and journaling in a storage device, and recovering the storage device, as recited in these claims.

Appellant provides additional reasons for the traversal of the second ground of rejection with regard to specific claims below.

Claims 13 and 30:

Claims 13 and 30 depend from claims 12 and 29, respectively, and thus the rejection of claims 13 and 30 is in error for at least the reasons given above for claims 12 and 29. Additionally, each of claims 13 and 30 recite a combination of features including: "writing a master inode corresponding to an inode file including said second inode to a checkpoint record in said journal."

The Office Action alleges that the above highlighted features are taught in Raz at col. 62, line 3-col. 63, line 44. Specifically, the Office Action alleges that Raz's after-image log teaches the above highlighted features. While Raz does teach writing records to the after-image log and flushing the updates to state memory at certain checkpoints, nothing in Raz teaches or suggests a master inode corresponding to an inode file that includes the second inode, nor writing the master inode to a checkpoint record.

For at least all of the above reasons, Appellant respectfully submits that the rejection of claims 13 and 30 is in error and requests reversal of the rejection.

Claims 14 and 31:

Claims 14 and 31 depend from claims 13 and 30, respectively, and thus the rejection of claims 14 and 31 is in error for at least the reasons given above for claims 13 and 30. Additionally, each of claims 14 and 31 recite a combination of features including: "copying said master inode from said most recent checkpoint record to a volatile memory".

As highlighted above, Raz does not teach or suggest writing a master inode to a checkpoint record. For similar reasons, Raz also does not teach or suggest "copying said master inode from said most recent checkpoint record to a volatile memory".

For at least all of the above reasons, Appellant respectfully submits that the rejection of claims 14 and 31 is in error and requests reversal of the rejection.

Claims 15 and 32:

Claims 15 and 32 depend from claims 14 and 31, respectively, and thus the rejection of claims 15 and 32 is in error for at least the reasons given above for claims 14 and 31. Additionally, each of claims 15 and 32 recite a combination of features including: "updating said master inode in said volatile memory to point to said copied one or more blocks".

As highlighted above, Raz does not teach or suggest writing a master inode to a checkpoint record. For similar reasons, Raz also does not teach or suggest "updating said master inode in said volatile memory to point to said copied one or more blocks".

For at least all of the above reasons, Appellant respectfully submits that the rejection of claims 15 and 32 is in error and requests reversal of the rejection.

Claims 16 and 33:

Claims 16 and 33 depend from claims 12 and 29, respectively, and thus the rejection of claims 16 and 33 is in error for at least the reasons given above for claims 12 and 29. Additionally, each of claims 16 and 33 recite a combination of features including:

> copying said first inode allocation bitmap to a second inode allocation bitmap;
>
> modifying said second inode allocation bitmap to reflect one or more inodes allocated to new files; and
>
> establishing a third inode within said block map to said second inode allocation bitmap subsequent to said modifying said second inode bitmap.

The features of claims 16 and 33 do not appear to be explicitly treated in the Office Action. Accordingly, **a proper *prima facie* case of obviousness has not been established** with regard to claims 16 and 33. Appellant further submits that none of the above highlighted features are taught by Raz's database.

For at least all of the above reasons, Appellant respectfully submits that the rejection of claims 16 and 33 is in error and request reversals of the rejection.

Claims 17 and 34:

Claims 17 and 34 depend from claims 16 and 33, respectively, and thus the rejection of claims 17 and 34 is in error for at least the reasons given above for claims 16 and 33. Additionally, each of claims 17 and 34 recite a combination of features including:

> copying said first block allocation bitmap to a second block allocation bitmap;
>
> modifying said second block allocation bitmap to reflect one or more blocks allocated to files; and
>
> establishing a fourth inode within said block map to said second block allocation bitmap subsequent to said modifying said second block allocation bitmap.

The features of claims 17 and 34 do not appear to be explicitly treated in the Office Action. Accordingly, **a proper *prima facie* case of obviousness has not been established** with regard to claims 17 and 34. Appellant further submits that none of the above highlighted features are taught by Raz's database.

For at least all of the above reasons, Appellant respectfully submits that the rejection of claims 17 and 34 is in error and requests reversal of the rejection.

Third Ground of Rejection:

Claims 40-44 are rejected under 35 U.S.C. § 102(b) as being anticipated by Senator. Appellant respectfully traverses the rejection for at least the reasons set forth below.

Claims 40-42 and 44:

Appellant respectfully submits that each of claims 40-42 and 44 recite a combination of features not taught or suggested in Senator. For example, claim 40 recites a combination of features including:

> a computing node configured to generate a plurality of write commands to update a first file and a commit command defined to commit the updates to the first file;
>
> an interconnect to which the computing node is coupled, wherein the computing node is configured to transmit the plurality of write commands and the commit command on the interconnect; and
>
> a storage coupled to the interconnect and configured to store the first file, wherein the storage is configured to atomically update the first file to reflect the plurality of write commands responsive to the commit command.

Senator teaches a set of software modules (e.g. the reap module, rollback module, and version module) which execute on a host system. These modules communicate with the storage devices via the I/O system (for UFS) or the network I/O system (for NFS). That is, the modules are on the same side of the I/O systems as the user applications and other software that cause writes to files. Senator teaches: "The VERSION module 107 provides for various versions of a file without the need for data copy or log operations. Module 107 receives an input signal from either the COMMIT system call or the FSYNC call. The ROLLBACK module 109 restores a previous version of a file by associating the file name with the metadata for the previous version. The REAP module removes previous versions of a file that are determined to be no longer needed and impractical to retain indefinitely." Accordingly, these modules handle the creation and deletion of file versions. None of this teaches or suggests: "a computing node configured to generate a

plurality of write commands ...an interconnect to which the computing node is coupled ...and a storage coupled to the interconnect and configured to store the first file, wherein the storage is configured to atomically update the first file to reflect the plurality of write commands responsive to the commit command."

For at least all of the above reasons, Appellant respectfully submits that the rejection of claim 40 is in error and request reversal of the rejection. The rejection of claims 41-42 and 44, being dependent from claim 40, is also in error and reversal thereof is requested. Each of claims 41-42 and 44 recite additional combinations of features not taught or suggested in the cited art.

Claim 43:

Claim 43 depends from claim 40, and thus the rejection of claim 43 is in error for at least the reasons given above for claim 40. Additionally, claim 43 recites a combination of features including: "the storage is an object-based storage and wherein the plurality of write commands and the commit command address a file object corresponding to the first file."

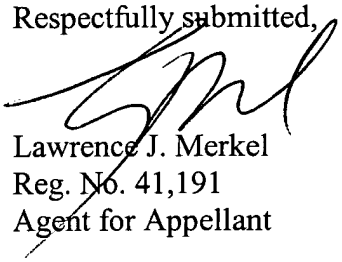
The Office Action alleges that Senator teaches the above highlighted features. However, nothing in Senator teaches an object-based storage. For at least this additional reason, Appellant respectfully submits that the rejection of claim 40 is in error and request reversal of the rejection.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejections of claims 2-10, 12-20, 22-27, 29-35, and 40-44 under 35 U.S.C. §§ 102(b) and 103(a) are erroneous, and reversal of the decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-59100/LJM. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,



Lawrence J. Merkel
Reg. No. 41,191
Agent for Appellant

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850

Date: March 13, 2006

IX. CLAIMS APPENDIX

The claims on appeal are as follows.

2. A storage comprising:

a non-volatile memory storing a first inode locating a first file in said storage and also storing a journal comprising a list of committed inodes; and

a block manager configured to copy said first inode to a second inode, wherein said block manager is configured to change said second inode in response to updates to said first file, and wherein said block manager is configured to atomically update said first file in response to a commit of said first file by writing said second inode to said non-volatile memory, whereby said second inode locates said first file in said storage, and wherein said block manager is configured to record said second inode in said journal.

3. The storage as recited in claim 2 wherein said commit of said first file comprises a commit command received from an external source which updates said first file.

4. The storage as recited in claim 3 wherein said commit command comprises a file close command.

5. The storage as recited in claim 3 wherein said commit command comprises an fsync command.

6. The storage as recited in claim 2 wherein said journal further includes a checkpoint record including a description of an inode file, a block allocation bitmap, and an inode allocation bitmap.

7. The storage as recited in claim 6 wherein the description comprises inodes for each of

said inode file, said block allocation bitmap, and said inode allocation bitmap.

8. An apparatus comprising:

a computing node configured to perform one or more write commands to a file and a commit command committing the one or more write commands to said file; and

a storage coupled to receive said one or more write commands and said commit command, wherein said storage is configured to copy one or more blocks of said file to a copied one or more blocks, said one or more blocks updated by said one or more write commands, and wherein said storage is configured to update said copied one or more blocks with write data corresponding to said one or more write commands, and wherein said storage is configured to copy a first inode locating said file in said storage to a second inode and to update pointers within said second inode pointing to said one or more blocks to point to said copied one or more blocks, and wherein said storage is configured to atomically update said file by writing said second inode responsive to said commit command, and wherein said first inode is stored in an inode file, and wherein said inode file is identified by a master inode, and wherein said inode file is atomically updated with said second inode by writing said master inode subsequent to said commit command.

9. The apparatus as recited in claim 6 wherein said commit command comprises a file close command.

10. The apparatus as recited in claim 6 wherein said commit command comprises an fsync command.

12. A method comprising:

copying a first inode to a second inode, wherein said first inode locates a first file in a storage;

modifying said second inode in response to one or more changes to said first file; and

atomically updating said first file by establishing said second inode as the inode for said first file, wherein said establishing comprises storing said second inode in a journal stored in a nonvolatile memory.

13. The method as recited in claim 12 further comprising writing a master inode corresponding to an inode file including said second inode to a checkpoint record in said journal.

14. The method as recited in claim 13 wherein recovering from a system failure comprises:

scanning said journal to locate a most recent checkpoint record and zero or more inodes subsequent to said most recent checkpoint record within said journal;

copying said master inode from said most recent checkpoint record to a volatile memory; and

updating an inode file corresponding to said master inode with said one or more inodes subsequent to said most recent checkpoint record.

15. The method as recited in claim 14 wherein said updating said inode file comprises:

copying one or more blocks of said inode file storing said one or more inodes to a

copied one or more blocks; and

updating said master inode in said volatile memory to point to said copied one or more blocks.

16. The method as recited in claim 12 wherein said block map further comprises a first inode allocation bitmap indicating which inodes within said first inode file are allocated to files, the method further comprising:

copying said first inode allocation bitmap to a second inode allocation bitmap;

modifying said second inode allocation bitmap to reflect one or more inodes allocated to new files; and

establishing a third inode within said block map to said second inode allocation bitmap subsequent to said modifying said second inode bitmap.

17. The method as recited in claim 16 wherein said block map further comprises a first block allocation bitmap indicating which blocks within a storage including said block map are allocated to files, the method further comprising:

copying said first block allocation bitmap to a second block allocation bitmap;

modifying said second block allocation bitmap to reflect one or more blocks allocated to files; and

establishing a fourth inode within said block map to said second block allocation bitmap subsequent to said modifying said second block allocation bitmap.

18. The method as recited in claim 12 wherein said establishing said second inode is performed in response to a commit command.

19. The method as recited in claim 18 wherein said commit command is a close file command.

20. The method as recited in claim 18 wherein said commit command is an fsync command.

22. A storage comprising:

a non-volatile memory storing a first inode locating a first version of a file in said storage and also storing a journal comprising a list of committed inodes; and

a block manager configured to copy said first inode to a second inode, wherein said block manager is configured to change said second inode in response to updates to the file, and wherein said block manager is configured to atomically update the file, producing a second version of the file, in response to a commit of the file by writing said second inode to said non-volatile memory, wherein said second inode locates said second version of said file in said storage, and wherein said block manager is configured to record said second inode in said journal.

23. The storage as recited in claim 22 wherein said commit of the file comprises a commit command received from an external source which updates the file.

24. The storage as recited in claim 23 wherein said commit command comprises a file close command.

25. The storage as recited in claim 23 wherein said commit command comprises an fsync command.

26. The storage as recited in claim 22 wherein said journal further includes a checkpoint record including a description of an inode file, a block allocation bitmap, and an inode allocation bitmap.

27. The storage as recited in claim 26 wherein the description comprises inodes for each of said inode file, said block allocation bitmap, and said inode allocation bitmap.

29. A method comprising:

copying a first inode to a second inode, wherein said first inode locates a first version of a file in a storage;

modifying said second inode in response to one or more changes to the file, creating a second version of the file; and

atomically updating the file to the second version by establishing said second inode as the inode for the file, wherein said establishing comprises storing said second inode in a journal stored in a nonvolatile memory.

30. The method as recited in claim 29 further comprising writing a master inode corresponding to an inode file including said second inode to a checkpoint record in said journal.

31. The method as recited in claim 30 wherein recovering from a system failure comprises:

scanning said journal to locate a most recent checkpoint record and zero or more inodes subsequent to said most recent checkpoint record within said journal;

copying said master inode from said most recent checkpoint record to a volatile

memory; and

updating an inode file corresponding to said master inode with said one or more inodes subsequent to said most recent checkpoint record.

32. The method as recited in claim 31 wherein said updating said inode file comprises:

copying one or more blocks of said inode file storing said one or more inodes to a copied one or more blocks; and

updating said master inode in said volatile memory to point to said copied one or more blocks.

33. The method as recited in claim 29 wherein said block map further comprises a first inode allocation bitmap indicating which inodes within said first inode file are allocated to files, the method further comprising:

copying said first inode allocation bitmap to a second inode allocation bitmap;

modifying said second inode allocation bitmap to reflect one or more inodes allocated to new files; and

establishing a third inode within said block map to said second inode allocation bitmap subsequent to said modifying said second inode bitmap.

34. The method as recited in claim 33 wherein said block map further comprises a first block allocation bitmap indicating which blocks within a storage including said block map are allocated to files, the method further comprising:

copying said first block allocation bitmap to a second block allocation bitmap;

modifying said second block allocation bitmap to reflect one or more blocks allocated to files; and

establishing a fourth inode within said block map to said second block allocation bitmap subsequent to said modifying said second block allocation bitmap.

35. The method as recited in claim 29 wherein said establishing said second inode is performed in response to a commit command.

40. An apparatus comprising:

a computing node configured to generate a plurality of write commands to update a first file and a commit command defined to commit the updates to the first file;

an interconnect to which the computing node is coupled, wherein the computing node is configured to transmit the plurality of write commands and the commit command on the interconnect; and

a storage coupled to the interconnect and configured to store the first file, wherein the storage is configured to atomically update the first file to reflect the plurality of write commands responsive to the commit command.

41. The apparatus as recited in claim 40 wherein the first file on the storage prior to the plurality of write commands is a first version of the first file, and wherein the storage is configured to create a second version of the first file in response to a first write command of the plurality of write commands, and wherein the storage is configured to atomically replace the first version with the second version in response to the commit command.

42. The apparatus as recited in claim 41 wherein the first version is associated with a first inode, and wherein the second version is created by copying the first inode to a

second inode and modifying the second inode, and wherein the atomic update is performed by writing the second inode.

43. The apparatus as recited in claim 40 wherein the storage is an object-based storage and wherein the plurality of write commands and the commit command address a file object corresponding to the first file.

44. The apparatus as recited in claim 40 further comprising a second computing node coupled to the interconnect, wherein the second computing node is configured to generate a second plurality of write commands to update the first file and a second commit command defined to commit the updates to the first file, and wherein the second computing node is configured to transmit the second plurality of write commands and the second commit command on the interconnect, and wherein the storage is configured to atomically update the first file to reflect the second plurality of write commands responsive to the second commit command.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings known to Appellant.